УДК: 519.1:519.7:007.5 https://doi.org/10.15407/jai2025.01.027

Стьопкін А. В.
ДВНЗ «Донбаський державний педагогічний університет», Україна
вул. Наукова, 13, м. Дніпро, 49050
stepkin.andrej@gmail.com
https://orcid.org/0000-0002-6130-9920

МУЛЬТИАГЕНТНА СИСТЕМА РОЗПІЗНАВАННЯ ГРАФІВ

Анотація. В статті розглядається проблема розпізнавання простих неорієнтованих графів мультиагентними системами. Система, що розглядається в статті, складається з трьох агентів: два агенти-дослідники, які можуть рухатись графом, зчитувати та змінювати помітки елементів графа та обмінюватись інформацією з третім агентом – агентом-експериментатором, який і будує мапу досліджуваного графу в своїй пам'яті у вигляді списків ребер та вершин. Запропоновано алгоритм розпізнавання квадратичної (від числа вершин графа) часової, ємнісної та комунікаційної складностей. Число переходів по ребрах, які треба здійснити агентам-дослідникам, оцінюється як $O(n)$. Також у роботі наведено процедуру пошуку нових підграфів для розпізнавання, у випадку, коли один з агентів-дослідників закінчив розпізнавання своєї частини, а другий продовжує роботу, що дозволяє зменшити витрати часу та більш рівномірно використовувати ресурси агентів-дослідників для розпізнавання графа. Для роботи алгоритму використовується три фарби різного кольору. Метод базується на методі обходу графа у глибину.

Ключові слова: розпізнавання графа, мультиагентна система, обхід графа, складність алгоритму, обхід в глибину.

Stopkin A.
SHEI "Donbas State Pedagogical University", Ukraine
13, Naukova st., Dnipro, 49050
stepkin.andrej@gmail.com
https://orcid.org/0000-0002-6130-9920

MULTI-AGENT GRAPH EXPLORATION SYSTEM

Abstract. The article considers the problem of simple undirected graphs exploration by multi-agent systems. The system considered in the article consists of three agents: two agents-researchers, that can move through the graph, read and change the labels of the graph elements, and exchange information with the third agent - the agent-experimenter, which builds a map of the explored graph in its memory. An algorithm for exploration of quadratic (from the number of graph nodes) time, space and communication complexities is proposed. The number of transitions along the edges that should be made by agents-researchers is estimated as $O(n)$. The paper also provides a procedure for finding new subgraphs for exploration, in the case when one of the agents-researchers has finished exploring its part, and the second one continues to work. That allows to reduce time consumption and more evenly use the resources of agents-researchers for graph exploration. The algorithm uses three different colors. The method is based on the depth-first graph traversal method.

Keywords: graph exploration, multi-agent system, graph traversal, algorithm complexity, depth-first traversal method.

Вступ

Враховуючи той факт, що в світі існує величезна кількість різноманітних недосліджених середовищ [1], активного розвитку набуває такий напрям математичної кібернетики як теорія дискретних динамічних систем. А так як дослідження середовища – це свого роду дискретна система, то це призвело до інтенсивного дослідження поведінки автоматів в лабіринтах [2]. Також в наш час активно вивчаються особливості функціонування різних мультиагентних систем [3,4]. В контексті вивчення таких систем виділяють три основні напрями: керування колективом агентів [5], формація колективу агентів [6] та консенсус колективу агентів [7]. Таким чином дослідження мультиагентних систем розпізнавання графів є досить актуальним питанням.

Постановка проблеми

Під розпізнаванням графів розуміємо проблему побудови мапи досліджуваного графа. Тобто створення маршрутів руху

агентів по невідомому графу, розмітки його елементів, збору та обробки локальної інформації про граф, яка дозволить побудувати мапу графа, з точністю до позначок на його елементах. Також досить важливим є завдання оптимізації витрат часу та ресурсів для розпізнавання графів. Зрозуміло, що при розпізнаванні графа колективом агентів, що блукають по ньому, основною проблемою є проблема ефективності їх взаємодії, з метою зменшення витрат часу і пам’яті на розпізнавання. Тобто при вирішенні проблеми розпізнавання графів мультиагентною системою необхідно розробити такі алгоритми функціонування, в яких блукаючі агенти не заважають один одному і не дублюють роботу один одного.

Аналіз останніх досліджень і публікацій

Вивчення алгоритмів обходу графів започаткував К. Шеннон в своїй роботі [8], в якій розглядалася задача пошуку автоматом заданої цілі в лабіринті. В подальшому вивченням особливостей функціонування автоматів в шахових лабіринтах в своїх дослідженнях займався К. Депп [9]. Дослідженням розпізнавання графів активно займалася група вчених на чолі з Г. Дудеком [10]. В їх роботах проводились дослідження невідомих середовищ при різній апріорній інформації про них. Також активно вивчалися мультиагентні системи по типу роя. [11,12]. Також широко досліджувалися алгоритми роботи мультиагентних систем, в яких обмін інформацію між блукаючими агентами відбувався за допомогою міток на елементах графа [13-15].

Мета дослідження

Метою нашого дослідження є створення ефективного методу і побудова алгоритму розпізнавання простих неорієнтованих графів мультиагентною системою, ефективність якого або перевищить раніше побудовані алгоритми, або зменшить кількість використовуваних для роботи ресурсів [3,15]. Таким чином дослідження часової, ємнісної, комунікаційної складностей побудованого алгоритму та кількості переходів по ребрах, які необхідно виконати агентам-дослідникам для розпізнавання графа також є важливою частиною нашого дослідження.

Виклад основного матеріалу

Нехай $G=(V,E)$ – простий зв'язний неорієнтований скінчений граф без петель та кратних ребер, де V – множина вершин графа, E – множина ребер (двохелементних підмножин (u,v), де $u,v\in V$) графа. Трійку $((u,v),v)$ називатимемо інцидентором (точкою дотику) ребра (u,v) та вершини v. Через I позначимо множину всіх інциденторів графа. Множину $L=V\cup E\cup I$ назвемо множиною елементів графа G. Сюр'єктивне відображення $\mu:L\rightarrow\{w,r,y,ry,b\}$, де w інтерпретується як білий колір, r – червоний, y – жовтий, ry – червоно-жовтий, b – чорний, назвемо функцією розмальовки графа G. Пара (G,μ) називається розфарбованим графом. Послідовність попарно суміжних вершин графа називається шляхом довжини . Околом вершини будемо називати множину елементів графа, що складається з вершини , всіх вершин суміжних з , всіх ребер та всіх інцидентів . Потужність множини вершин і ребер позначимо через і відповідно. Зрозуміло, що . Ізоморфізмом графа та графа назвемо таку бієкцію , що тоді, й тільки тоді, коли . Таким чином, ізоморфні графи рівні з точністю до відміток на елементах графа. Під зворотними ребрами ми будемо розуміти білі ребра, в яких поточна та дальня вершина забарвлена в «свій» колір. Під перешийками будемо розуміти ребра, які з’єднують підграфи роботи різних агентів.

Мультиагентна система, що розпізнає граф , складається з трьох агентів: два агенти-дослідники і та один агент-експериментатор. Агенти-дослідники

переміщаються по графу і можуть змінювати забарвлення елементів графа. Агент-експериментатор – нерухомий агент, який знаходить поза межами графа та на основі повідомлень від агента-дослідника будує мапу графа в своїй пам'яті у вигляді списків вершин та ребер. Агенти-дослідники мають скінчену пам'ять, яка може вмістити номер вершини. На початку роботи агенти A і B поміщаються в довільні неспівпадаючі вершини графа G, одразу нумерують їх і передають номери агенту-експериментатору, який поміщає їх у множину вершин V_H. Агенти пересуваються по графу G з вершини v в вершину u по ребру (v,u), можуть змінювати забарвлення вершин, ребер та інцидентoрів, а також записувати номери у вершини графа. Перебуваючи у вершині v, агент-дослідник сприймає мітки всіх елементів околу $Q(v)$ та номери суміжних з нею вершин. Виходячи з цієї інформації, визначає, яким ребром буде далі рухатись та як фарбуватиме елементи графа, через які відбувався рух. Агент-експериментатор може як передавати, так і приймати повідомлення. Він володіє скінченою на кожному кроці, але необмежено зростаючою внутрішньою пам'яттю, в якій фіксується результат функціонування агентів-дослідників на кожному кроці. Зазначимо, що агенти-дослідники обмінюються повідомленнями тільки з агентом-експериментатором, один з одним вони взаємодіють виключно за рахунок відміток на елементах графа.

Розглянемо детально режими роботи агентів-дослідників.

До початку роботи мультиагентної системи всі елементи графа мають білий колір. При описі режимів роботи агентів, у дужках будемо вказувати повідомлення, які відправляють агенти-дослідники агенту-експериментатору в конкретному випадку (MESSAGE_A; MESSAGE_B). Після відправки таких повідомлень агент-експериментатор обробляє їх та відправляє назад агентам-дослідникам дані, необхідні для завершення ходу. Процедури обробки повідомлень ми розглянемо нижче окремим блоком.

Звичайний режим роботи. Агент-дослідник сканує окіл вершини, в якій він знаходиться, на наявність білих вершин, рахує кількість білих вершин, що залишаться в околі вершини після того, як агент-дослідник перейде з неї. Далі агент обирає довільну білу вершину і переходить в неї (FORWARD_A($x_1, x_2,..., x_q$); FORWARD _B($k_1, k_2,..., k_g$)), де x_i, k_i - номер вершини, з якої було зроблено перехід, а q, g – кількість білих вершин, що залишилися в її околі, після відходу з неї агента. Таким чином, агент-дослідник рухається вперед по білих вершинах, фарбуючи вершини, ребра, що їх з'єднують та дальні інцидентори в свій колір. Відвідуючи нові вершини, агент-дослідник записує в них номери, отримані від агента-експериментатора.

Якщо в околі вершини, в якій знаходиться агент-дослідник, не знайдеться білої вершини для подальшого руху вперед, то агент A відправляє агенту-експериментатору повідомлення (ASSIGN_A), на що витрачає один хід. Далі агент повертається назад, забарвлюючи пройдені вершини, ребра та ближні інцидентори в чорний колір (BACK_A). Агент-дослідник B, не виявивши білої вершини для подальшого руху вперед, відразу починає рух назад, забарвлюючи пройдені вершини, ребра та ближні інциденти у чорний колір (BACK_B). Повертаючись назад своїм шляхом, агенти-дослідники на кожному кроці перевіряють наявність білих ребер з дальньою вершиною, пофарбованою в «чужий» колір (перешийків). Якщо така вершина існує, але другим агентом ще не позначено перший перешийок, то агент-дослідник, що розглядається, пропускатиме хід, поки другий агент не помітить свій перший перешийок. Повернувшись до початкової вершини, агент-дослідник завершує роботу на даній території (STOP_A; STOP_B). Таким чином, будується кілька (не менше двох) дерев шляхом обходу в глибину.

Закінчивши роботу на своїй території (всі вершини цієї території забарвлені у чорний колір), агент-дослідник шукатиме

нову територію для розпізнавання. Тому, під час розпізнавання, йому необхідно, якимось чином позначити шлях до вершини (назвемо її вершиною переходу) інцидентного перешийку, через який буде здійснено перехід. Шлях до вершини переходу, зрештою, складатиметься з ребер (v,u), пофарбованих

$$(\mu((v,u),v)=w) and (\mu(v,u)=b) and$$
$$and (\mu((v,u),u)=r) and (\mu(u)=b) \text{ для агента } A$$
$$\text{та } (\mu((s,z),s)=w) and (\mu(s,z)=b) and$$
$$and (\mu((s,z),z)=y) and (\mu(z)=b) \text{ для агента } B.$$

Розглянемо можливі ситуації та особливості функціонування агентів-дослідників у кожній із ситуацій. Звернемо увагу, що, називаючи нижче агентів «першим» і «другим», розуміємо, що перший агент (їм може бути будь-який з агентів-дослідників) – це агент, через територію якого відбувається перехід, а другий – агент, який здійснює перехід через чужу область.

1. Побудова шляху до першого, виявленого на своєму шляху перешийку.

У цьому випадку зміни в звичайному режимі роботи агентів торкнуться тільки способу фарбування і відбудуться лише тоді, коли агент-дослідник переходитиме з вершини з номером меншим (або рівним) номеру вершини переходу даного агента-дослідника, у вершину з меншим номером. У такому разі, при русі назад своїм шляхом, агент-дослідник забарвлює пройдені вершини і ребра в чорний колір (BACK_A; BACK_B), ближній інцидентор при цьому не зафарбовується і залишається зафарбованим у «свій» колір. Надалі, за такими інциденторами агент-дослідник і орієнтуватиметься під час руху до перешийка для переходу в нову область розпізнавання.

При переході одного з агентів-дослідників в новий нерозпізнаний підграф, частина графа в іншого агента-дослідника змінюється. Фіксується (якщо раніше не виявлено жодного перешийка) перешийок для переходу в чужу область розпізнавання (зазначимо, що його вершина переходу не зміниться тільки в тому випадку, коли новий і старий перешийки інцидентні одній вершині). Це вимагатиме модифікації старого шляху, або побудови нового шляху для переходу.

2. Побудова агентом-дослідником шляху до нового перешийка, отриманого після переходу іншого агента-дослідника в нову частину графа, що вимагає розпізнавання.

2.1. Якщо до призначення нової вершини переходу агент-дослідник не виявив жодного перешийка, то побудова нового шляху відбувається аналогічно до побудови першого.

2.2. Якщо нова вершина для переходу знаходиться в тій же гілці дерева, що і вершина переходу, тільки глибше, то зміни в звичайному режимі роботи агента-дослідника будуть такі ж, як і при побудові шляху до першого, виявленого на своєму шляху перешийка, тільки настануть, починаючи з моменту потрапляння агента-дослідника в нову вершину переходу.

2.3. Якщо при зміні перешийка для переходу нова вершина переходу збігається з вершиною переходу, то зміни в роботі агента-дослідника будуть такі ж, як і при побудові шляху до першого, виявленого на своєму шляху перешийка.

2.4. Якщо нова вершина переходу знаходиться в тій самій гілці дерева, що і вершина переходу, тільки ближче до вихідної вершини (номер нової вершини переходу менший за номер вершини переходу), то агент-дослідник, потрапивши в нову вершину переходу, забарвлює ближній інцидентор ребра, що продовжує старий шлях у «свій» колір (SHORT_PATH_A, SHORT_PATH_B). Тим самим скорочуючи старий шлях, ми отримуємо шлях до нової вершини переходу. Далі агент-дослідник функціонує так само, як і при побудові шляху до першого, виявленого на своєму шляху перешийка.

2.5. Якщо нова вершина переходу знаходиться у гілці дерева, відмінної від тих, в яких знаходяться вершини переходу та/або попередні нові вершини переходу (якщо такі є), то агент-дослідник, потрапивши в нову вершину переходу, починає функціонувати, як і при побудові шляху до першого, виявленого на своєму шляху перешийка. До того моменту, поки

не потрапить у вершину, з якої беруть початок гілки, що містять вершину переходу (або попередню нову вершину переходу) та нову вершину переходу. У цій вершині агент-дослідник відправляє агенту-експериментатору номери дальніх вершин ребер, забарвлених у чорний колір з дальнім інцидентом, забарвленим у «свій» колір (SEND_NUMB_A(x_1, x_2); SEND_NUMB_B(k_1, k_2)). Агент-експериментатор в свою чергу визначає менший з них, запам'ятовує його. Далі агент-дослідник вибирає довільне з цих двох ребер, забарвлює ближній інцидентор у «свій» колір і відправляє агенту-експериментатору номер дальньої вершини, обраного ребра (SELECT_NODE_A(x),SELECT_NODE_B (k)). Агент-експериментатор в свою чергу порівнює номер отриманої вершини з мінімальним номером, збереженим раніше. Якщо номери збіглися, то інцидентор забарвлений правильно, старий шлях розірвано (RESET_TRANSIT_A, RESET_TRANSIT_B). Якщо ж номери не збіглися, то пофарбований на минулому кроці інцидентор забарвлюється в білий колір, а ближній інцидент другого такого ребра забарвлюється в «свій» колір, тим самим розриваючи старий шлях. Агент-дослідник діє подібним чином, доки не обріже останній старий шлях, який веде до вершини переходу або попередніх нових вершин переходу. Далі агент-дослідник функціонує так само, як і при побудові шляху до першого, виявленого на своєму шляху перешийка.

Зазначимо, що агент-дослідник за результатами сканування точно визначити чи знаходиться він в іншій гілці, чи ні, не може і визначення відбувається лише за номерами вершин. Це може спричинити ситуацію, коли агент-дослідник шукатиме вершину, з якої беруть початок гілки, що містять вершину переходу (або попередню нову вершину переходу) і нову вершину переходу, а їх просто не існує. Щоб виключити подібну ситуацію, агент-експериментатор відстежує номери вершин, які відвідує агент-дослідник. Коли номер вершини, в якій знаходиться агент-дослідник, більший за номер вершини переходу, то можливі два варіанти. *Агент-дослідник знаходиться в тій самій гілці, що й вершина переходу, тільки глибше.* У такому випадку, при поверненні назад по своєму шляху, агент-дослідник рано чи пізно потрапить у вершину переходу, й змінна one_branch_A (one_branch_B – для агента B), що зберігається агентом-експериментатором, приймає значення «1». Агент-дослідник, виявивши зміну значення відповідної змінної, відправляє агенту-експериментатору повідомлення (CANCEL_SHORT_PATH_A, CANCEL_SHORT_PATH_B). Це дозволить агентам-дослідникам не шукати вершину, з якої беруть початок гілки, що містять вершину переходу і нову вершину переходу, до потрапляння в наступну нову вершину переходу (якщо така з'явиться). *Агент-дослідник знаходиться в іншій гілці.* Тоді рано чи пізно агент-дослідник потрапить у вершину, з якої беруть початок гілки, що містять вершину переходу та нову вершину переходу. Тоді змінна $end_another_branch_A$ ($end_another_branch_B$ для агента B), що зберігається агентом-експериментатором, приймає значення «1», агент-дослідник, виявивши зміну відповідної змінної, відправляє агенту-експериментатору повідомлення («CANCEL_SHORT_PATH_A», «CANCEL_SHORT_PATH_B») і далі не шукає вершину, з якої беруть початок гілки, що містять вершину переходу та нову вершину переходу, до попадання у наступну нову вершину переходу (якщо така з'явиться).

3. Побудова першим агентом-дослідником шляху до нового перешийка, отриманого після більш ніж одного переходу другого агента-дослідника в новий підграф для розпізнавання, при незмінній області функціонування першого агента-дослідника.

У цьому випадку відбувається зміна перешийка для переходу в чужу область і відповідно отримуємо нову вершину переходу. Так як таке можливе тільки для вершин, що знаходяться в тій самій гілці, в якій в даний момент працює перший агент-дослідник і розташованих ближче, або на

такій же відстані до початкової вершини, як перший агент-дослідник, то при побудові нового шляху використовуються принципи роботи з пунктів 2.3 - 2.5.

Режим розпізнавання обернених ребер. Якщо під час руху вперед було виявлено зворотне ребро, то агент-дослідник сканує окіл поточної вершини, покроково зчитує номери всіх суміжних вершин, інцидентних зворотним ребрам, і відправляє список номерів агенту-експериментатору (INV_A($x_1,..., x_l$); INV_B($k_1,..., k_l$), де x_i, k_i– номери, записані агентами A та B відповідно, у вершинах свого шляху).

Режим розпізнавання перешийків. Цей режим дещо відрізняється для кожного з агентів-дослідників. Якщо, при русі вперед, агент A виявив у вершині перешийки (ідентифікація перешийків відбувається по дальній вершині, пофарбованій в «чужий» колір і ближньому інциденту, забарвленому в білий колір). При цьому в жодній з дальніх вершин цих перешийків немає агента B (або агент B знаходиться в одній з цих вершин, але вже розпізнав всі, раніше виявлені ним перешийки в цю вершину, або B виконує повернення назад своїм шляхом). Агент A покроково зчитує номери дальніх вершин, інцидентних перешийкам та відправляє їх агенту-експериментатору ISTHM_A ($x_1, x_2,..., x_l$), де x_i– номери, записані агентом B, у вершинах свого шляху. Якщо агент B знаходиться в одній з дальніх вершин перешийків і ще не розпізнавав інцидентні їй перешийки або не повертається назад по своєму шляху, то агент A відправляє агенту-експериментатору номери всіх дальніх вершин, виявлених перешийків, окрім номера вершини, де знаходиться агент B. Зауважимо, що агент A дізнається про знаходження B в суміжній вершині в результаті сканування околу на наявність перешийків, але про те, чи можна розпізнавати перешийок, в дальній вершині якого знаходиться B, агент A дізнається зі значення змінної mr_A, що запитується у агента-експериментатора. Якщо, при русі вперед, перешийки виявив агент B і в жодній з дальніх вершин цих перешийків немає агента A (або A знаходиться в одній з цих вершин, але виконує повернення назад своїм шляхом), то B передає агенту-експериментатору номери всіх дальніх вершин виявлених перешийків (ISTHM _B($k_1, k_2,..., k_l$), де k_i– номери, записані агентом A у вершинах свого шляху. Якщо ж агент Aзнаходиться в одній з дальніх вершин перешийків і не виконує повернення назад своїм шляхом, то B не виконує жодних дій до виходу A з околу вершини, в якій знаходиться B. Агент B дізнається про знаходження A в суміжній вершині в результаті сканування околу на наявність перешийків. Але про те, чи можна розпізнавати перешийки за наявності в дальній вершині одного з них агента A, агент B дізнається зі значення змінної mr_B, що запитується у агента-експериментатора.

Звернемо увагу, що якщо агент-дослідник вперше потрапляє у вершину, що містить перешийки, то для можливості надалі переходити на чужу територію в пошуках нових частин графа для розпізнавання, агент-дослідник повинен помітити один з перешийків, яким можливо буде здійснений перехід. Цей перешийок і є першим виявленим перешийком. Тепер розглянемо, як це відбувається. Після відправлення агенту-експериментатору номерів дальніх вершин виявлених перешийків він визначає мінімальний номер та запам'ятовує його. Далі агент-дослідник починає перебирати всі перешийки, які він розпізнав у поточній вершині. Він вибирає довільне біле ребро з білими ближнім і дальнім інциденторами, дальню вершину якого забарвлено в «чужий» колір (при цьому якщо в дальній вершині перешийка знаходиться інший агент-дослідник, то перевіряти цей перешийок чи ні агент дізнається зі значення змінних mr_A, mr_B для агентів A і B відповідно), забарвлює ближній інцидентор у «свій» колір і відправляє номер дальньої вершини агенту-експериментатору (SELECT_NODE_A(x); SELECT_NODE_B(k)», де x, k – номери, записані агентами B і A відповідно у

вершинах свого шляху). Агент-експериментатор, в свою чергу, порівнює номер отриманої вершини з мінімальним номером, збереженим раніше. Якщо номери не збіглися, то ближній інцидентор перешийка, що розглядається, забарвлюється в чорний колір. Якщо ж номери збіглися, то, нічого не забарвлюючи, агент-дослідник відправляє агенту-експериментатору повідомлення (FIRST_ISTHM_A, FIRST_ISTHM_B).

У випадку, коли агент-дослідник рухається назад своїм шляхом, і у нього немає пропущених білих ребер і ще не помічений перший перешийок, він починає шукати перший виявлений перешийок іншого агента. Ідентифікація такого перешийка відбувається по дальній вершині, забарвленій в «чужий» (або чорний) колір і дальнього інцидентора, пофарбованого в «чужий» колір. Виявивши цей перешийок, агент-дослідник забарвлює його ближній інцидентор у «свій» колір і відправляє повідомлення агенту-експериментатору FIRST_ISTHM_A, FIRST_ISTHM_B). Далі побудова шляху до цього перешийка проводиться так само, як і при побудові шляху до першого виявленого перешийка. Якщо ж виявлено перешийок з дальньою вершиною, забарвленою в «чужий» колір і дальнім інцидентором, забарвленим у «чужий» (або в білий) колір. І при цьому іншим агентом-дослідником ще не помічено перший перешийок (це може статися, якщо він робить перебір виявлених перешийків для визначення першого), то агент-дослідник зупиняється до позначки іншим агентом першого перешийка. Зазначимо, що, потрапивши в нову область розпізнавання, першим перешийком для переходу в новій області агент-дослідник обирає ребро, яким було здійснено перехід у нову область.

При одночасному попаданні двох агентів-дослідників в одну білу вершину, кожен агент забарвлює вершину наполовину, і вона стає червоно-жовтою. Агент B, на наступному кроці, відступає назад своїм шляхом, видаляючи мітки, залишені ним на попередньому кроці (видаляється фарба з ребра і ближнього інцидентора), і переключається в звичайний режим роботи (RETURN_B). Агент A бачить різнокольорову вершину як свою і при розпізнаванні забарвлює у чорний колір всю вершину.

При попаданні агентів-дослідників у ситуацію, коли у вершині можливий вибір відразу кількох режимів роботи, першим буде активовано режим розпізнавання перешийків, за ним режим розпізнавання зворотних ребер і тільки після нього звичайний режим роботи. Потрапляння двох агентів-дослідників в одну білу вершину в цьому списку не розглядається, оскільки така ситуація призведе до змін у роботі виключно агента B і в цей момент інші режими роботи для нього будуть недоступні.

Тепер розглянемо режим роботи агента-дослідника, коли він закінчив розпізнавання своєї території, але другий агент залишив пропущені білі ребра (за якими можуть бути досить великі нерозпізнані підграфи). Агент-дослідник переходить по чорних ребрах із ближнім білим інцидентором, дальнім інцидентором «свого» кольору та чорною дальньою вершиною доти, доки це можливо. Якщо такого ребра не виявиться, це означатиме, що агент-дослідник потрапив у вершину, з якої можливий перехід у чужу область розпізнавання (вершина переходу або остання нова вершина переходу). Потрапивши в таку вершину, агент переходить у чужу область розпізнавання через білий перешийок із ближнім інцидентором, забарвленим у «свій» колір (якщо перехід виконується через вершину переходу). Або через білий перешийок з дальнім інцидентом і дальньою вершиною, забарвленим у «чужий» колір (якщо перехід виконується через останню нову вершину переходу). Тут можливі два варіанти:

1. *При переході агент потрапив у чорну вершину.* У цьому випадку агент-дослідник пересувається чорними вершинами, вибираючи ребро, у якого ближній інцидентор забарвлений у чорний або «чужий» колір. При цьому, якщо ближні чорні інцидентори, то виконуючи перехід, агент нічого не забарвлює, але

якщо інциденти пофарбовані в «чужий» колір, то, виконуючи перехід, агент забарвлює їх у чорний колір. Як тільки дальня вершина такого ребра буде пофарбована в «чужий» колір, агент-дослідник перевіряє наявність у цій вершині першого агента-дослідника (якщо перший агент-дослідник стоїть у цій вершині, то другий агент-дослідник не виконує жодних дій до виходу першого агента з вершини) і переходить до цієї вершини. Якщо ця вершина залишається забарвленою в «чужий» колір, то далі агент-дослідник працює за алгоритмом, описаним нижче в пункті 2 для випадку попадання агента у вершину, пофарбовану в «чужий» колір, інакше продовжує рух чорними вершинами. Звернемо увагу, що, пересуваючись по чорних вершинах чужого шляху, агент-дослідник на кожному кроці перевіряє через агента-експериментатора умову закінчення роботи першого агента-дослідника (кількість пропущених білих ребер дорівнює нулю, і відправлена команда про закінчення розпізнавання території $stop_condition_A = 1$ ($stop_condition_B = 1$ для агента B), якщо вона виконана, то алгоритм закінчує роботу.

2. *При переході агент потрапив у вершину, забарвлену у «чужий» колір.* І тут відразу перевіряється наявність пропущених білих ребер в першого агента. Якщо їх немає, то перевіряється, чи не закінчив роботу перший агент. Якщо не закінчив, другий агент-дослідник не виконує жодних дій до появи пропущених білих ребер або до завершення роботи першим агентом-дослідником. Якщо закінчив, алгоритм завершує роботу. Якщо пропущені білі ребра є, то агенту потрібно зорієнтуватися, в якому напрямку рухатися далі.

Якщо пропущені білі ребра є у вершині, в якій агент-дослідник знаходиться, то по чужому шляху він не пересувається, а відразу переходить у нову область розпізнавання, забарвлюючи дальній інцидентор і вершину в «свій» колір, а також відправляє номер вершини, з якої здійснює перехід (MOVING_N_AREA_A(x), MOVING_N_AREA_B(k), де x, k – номери, записані агентами B і A відповідно, у вершинах свого шляху). У разі переходу в чужу область розпізнавання та одночасного попадання з першим агентом-дослідником в одну вершину, перший агент пропускає хід, під час якого другий агент переходить у нову область розпізнавання, забарвлюючи дальні інцидентори і вершину у «свій» колір (MOVING_N_AREA_A(x), MOVING_N_AREA_B(k), де x, k – номери, записані агентами B і A відповідно, у вершинах свого шляху).

Якщо у вершині, де знаходиться агент-дослідник, немає пропущених білих ребер, то агент-дослідник відправляє агенту-експериментатору номер цієї вершини (INIT_POSITION_A(x), INIT_POSITION_B(k), де x, k – номери, записані агентами B і A відповідно, у вершинах свого шляху). Агент-експериментатор, у свою чергу, перевіряє, чи є в дереві вершини з пропущеними білими ребрами, що знаходяться ближче до вихідної вершини першого агента-дослідника, ніж вершина, в якій знаходиться другий агент. Якщо такі вершини є, то агент-дослідник починає рух назад по чужому шляху, перевіряючи на кожному кроці наявність пропущених білих ребер, виявивши такі ребра агент-дослідник переходить по кожному з них в нову область розпізнавання, забарвлюючи дальній інцидентор і вершину в «свій» колір (MOVING_N_AREA_A(x), MOVING_N_AREA_B(k), де x, k – номери, записані агентами B і A відповідно, у вершинах свого шляху).

Якщо ж вершини з пропущеними білими ребрами є лише глибше по дереву, то агент-дослідник почне рухатися вперед, чужим шляхом, до найближчої вершини з пропущеними білими ребрами, на кожному кроці перевіряючи їх наявність. Якщо таких ребер немає, то агент-дослідник зупиняється до їх появи, або до завершення роботи алгоритму. Якщо при русі вперед по чужому шляху другий агент потрапляє з першим агентом в одну вершину, з якої є пропущені білі ребра, перший агент зупиняється, до переходу

другого агента в нову область розпізнавання. Досягши потрібної вершини, агент переходить в нову область розпізнавання, забарвлюючи дальній інцидентор і вершину в «свій» колір (MOVING_N_AREA_A(x), MOVING_N_AREA_B(k), де x, k – номери, записані агентами B і A відповідно, у вершинах свого шляху). Після переходу в нову область розпізнавання агент-дослідник заново починає виконання алгоритму обходу, але нумерація продовжується далі.

Виконуючи обхід графа, агенти A і B створюють відповідно червоний та жовтий шляхи. Розглянемо принцип побудови агентами шляху «свого» кольору. При русі в білу вершину червоний (жовтий) шлях подовжується, при русі назад своїм шляхом – коротшає. Якщо агент-дослідник повернувся у вершину, з якої почав обхід графа, а в її околі не було білих вершин, то агент забарвлює цю вершину в чорний колір. Алгоритм закінчує роботу, коли червоний та жовтий шляхи стають порожніми, а всі вершини чорними. Виконуючи обхід графа G, агенти створюють нумерацію відвіданих вершин. Перший раз відвідавши вершину, агент A забарвлює їх у червоний колір (агент B – у жовтий колір), записує в пам'ять вершини відповідний номер (отриманий від агента-експериментатора), рівний значенню змінної ct_A (ct_B для агента B). Розпізнавання графа G відбувається на основі створеної агентами-дослідниками нумерації шляхом побудови графа H ізоморфного графу G.

Розглянемо алгоритм роботи агента-експериментатора.

Зауважимо, що далі в роботі закінчення « $_A$ » та « $_B$ » в назвах змінних означає, що змінна використовується для роботи агентів A і B відповідно, якщо не зазначено інше.

Вхід: списки повідомлень M та N від агентів-дослідників.

Вихід: список вершин V_H та ребер E_H графа H, ізоморфного графу G.

Дані: V_H, E_H – списки вершин і ребер графа H, ізоморфного графу G. ct_A, ct_B – лічильники числа відвіданих вершин. $STOP_A$, $STOP_B$ – змінні, що використовуються агентами для сигналізації агенту-експериментатору про завершення розпізнавання своєї області. mr_A, mr_B – змінні, що набувають значень «1» або «0». Значення «1» змінної mr_A дозволяє агенту A розпізнавати і перевіряти перешийки в дальній вершині яких знаходиться агент B. Значення "0" дозволяє агенту A розпізнавати і перевіряти тільки ті перешийки, в дальній вершині яких немає агента B. Значення «1» змінної mr_B дозволяє агенту розпізнавати та перевіряти всі перешийки, навіть за наявності в дальній вершині одного з перешийків агента A. Значення ж «0» забороняє агенту розпізнавання та перевірку тих перешийків, у дальній вершині яких знаходиться агент A. $r(1), r(2), \ldots, r(t)$ – список номерів вершин червоного шляху, де t - довжина цього списку. $y(1), y(2), \ldots, y(p)$ – список номерів вершин жовтого шляху, де p - довжина цього списку. Mes – поточне повідомлення. $first_isthm_A$, $first_isthm_B$ – змінні, що використовуються агентами для сигналізації агенту-експериментатору про те, що відповідний агент зафіксував перший перешийок. min_match_A, min_match_B – змінні, що використовуються агентами для визначення перешийка, інцидентного вершині з найменшим номером. Коли агент-дослідник вперше потрапляє у вершину з перешийками, він покроково сканує окіл поточної вершини та відправляє агенту-експериментатору номери всіх дальніх вершин, інцидентних перешийкам. Агент-експериментатор визначає мінімальний номер. Далі, агент-дослідник перебирає перешийки, відправляючи номер дальньої вершини обраного перешийка агенту-експериментатору. Як тільки відправлений номер збігається з мінімальним, відповідна змінна набуває значення «1». Після чого агент-дослідник фарбує необхідний нам

перешийок. $shorten_path_A$, $shorten_path_B$ – змінна, що сигналізує про те, що необхідно перебудувати шлях до нової вершини переходу. Розглянута змінна використовується в тому випадку, коли нова вершина переходу знаходиться з вершиною переходу в одній гілці, але вище по дереву. $shorten_path_AA$, $shorten_path_BB$ – змінна, що сигналізує про необхідність перебудувати шлях до нової вершини переходу. Розглянута змінна використовується в тому випадку, коли номер нової вершини переходу більший за номер вершини переходу. q_A, q_B – змінні для підрахунку кількості пропущених білих ребер на своєму шляху. Значення $back_color_A = 1$, $back_color_B = 1$ сигналізують агентам-дослідникам про те, що при русі назад ближній інцидентор не потрібно фарбувати в чорний колір і він залишається пофарбованим у «свій» для даного агента колір. У той час, як значення «0» говорить, що при кроці назад, ближній інцидентор потрібно зафарбувати в чорний колір. $stop_condition_A$, $stop_condition_B$ – змінні, що сигналізують агентам про те, що другий агент-дослідник закінчив розпізнавання своєї території і знаходиться в пошуку нової території для розпізнавання. $direction_A$, $direction_B$ – змінні, що визначають напрям руху по чужому шляху. Значення "1" – напрям назад, "0" – вперед. $new_trans_node_A$, $new_trans_node_B$ – змінні, в яких зберігається номер нової вершини переходу. $trans_node_A$, $trans_node_B$ – змінні, в яких зберігається номер вершини переходу. $test_node_A$, $test_node_B$ – змінні, що використовуються агентами під час руху назад своїм шляхом, для зберігання номера вершини, що належить шляху до вершини переходу, з якої є білий шлях. Тобто, коли агент-дослідник робить крок вперед з такої вершини, ця змінна приймає значення номера вершини, з якої почався рух вперед. На основі цих змінних формуються значення змінних $back_color_A$, $back_color_B$ відповідно. $chan_test_node_A$, $chan_test_node_B$ – змінні, що використовуються агентами для управління процесом зміни значень змінних $test_node_A$, $test_node_B$ відповідно. one_branch_A, one_branch_B – змінні, що використовуються для визначення місцезнаходження нової вершини переходу відносно вершини переходу. Якщо ці вершини знаходяться в одній гілці (що визначається попаданням агента-дослідника у вершину переходу, після руху назад, своїм шляхом, з нової вершини переходу), то змінній присвоюється значення «1», значення «0» – в іншому випадку. $end_another_branch_A$, $end_another_branch_B$ – змінні, що використовуються для визначення місцезнаходження нової вершини переходу відносно вершини переходу. Якщо ці вершини знаходяться в різних гілках (що визначається попаданням агента-дослідника у вершину з меншим номером, ніж у вершини переходу (без проходу через вершину переходу), після руху назад, своїм шляхом, з нової вершини переходу), то змінній присвоюється значення «1», значення «0» в іншому випадку. N_v_A, N_v_B – множини, що використовуються агентами для зберігання номерів вершин, у яких є пропущені білі ребра. Кількість однакових номерів у множині дорівнює кількості пропущених білих ребер з вершини з цим номером. $\min_n_A$, $\min_n_B$ – змінні, що використовуються агентами для зберігання мінімального номера вершини, необхідного при фарбуванні першого перешийка та відрізання шляху до вершини переходу або попередньої нової вершини переходу. $previous_back_A$, $previous_back_B$ – змінні, що використовуються агентами для визначення чи рухався агент-дослідник назад своїм шляхом на попередньому кроці.

1. $ct_A := 1$, $ct_B := 1$, $M := \varnothing$, $N := \varnothing$, $E_H := \varnothing$, $STOP_A := 0$, $STOP_B := 0$, $mr_A := 0$, $mr_B := 0$, $t := 1$, $p := 1$, $r(t) := ct_A$, $y(p) := ct_B$, $V_H := \{A[1], B[1]\}$, $first_isthm_A := 0$, $first_isthm_B := 0$, $\min_match_A := 0$, $\min_match_B := 0$, $shorten_path_A := 0$, $shorten_path_B := 0$, $shorten_path_AA := 0$, $shorten_path_BB := 0$, $q_A := 0$, $q_B := 0$, $back_color_A := 0$, $back_color_B := 0$, $stop_condition_A := 0$, $stop_condition_B := 0$, $direction_A := 0$, $direction_B := 0$, $new_trans_node_A := 0$, $new_trans_node_B := 0$, $trans_node_A := 0$, $trans_node_B := 0$, $test_node_A := 0$, $test_node_B := 0$, $chan_test_node_A := 0$, $chan_test_node_B := 0$, $one_branch_A := 0$, $one_branch_B := 0$, $end_another_branch_A := 0$, $end_another_branch_B := 0$, $N_v_A := \varnothing$, $N_v_B := \varnothing$, $\min_n_A := 0$, $\min_n_B := 0$, $previous_back_A := 0$, $previous_back_B := 0$;
2. while $(STOP_A = 0) or (STOP_B = 0)$ do
3. if $M \neq \varnothing$ then do
4. прочитати в Mes повідомлення та видалити його зі списку M;
5. LIST_PROCESS _ A ();
6. end do;
7. if $N \neq \varnothing$ then do
8. прочитати в Mes повідомлення та видалити його зі списку N;
9. LIST_PROCESS _B ();
10. end do;
11. if $(q_B = 0) and (STOP_B = 1)$ then $stop_condition_A := 1$;
12. if $(q_A = 0) and (STOP_A = 1)$ then $stop_condition_B := 1$;
13. if $(trans_node_A = r(t)) and (new_trans_nod_A = 0)$ then $test_node_A := r(t)$;
14. if $(trans_node_B = y(p)) and (new_trans_node_B = 0)$ then $test_node_B := y(p)$;
15. if $(new_trans_node_A = r(t)) and (new_trans_node_A < trans_node_A)$ t $shorten_path_A := 1$, $proveroch_versh_A := r(t)$;
16. if $(new_trans_node_B = y(p)) and (new_trans_node_B < trans_node_B)$ t $shorten_path_B := 1$, $proveroch_versh_B := y(p)$;
17. if $(new_trans_node_A = r(t)) and (new_trans_node_A = trans_node_A)$ then $shorten_path_A := 0$, $shorten_path_AA := 0$, $test_node_A := r(t)$;
18. if $(new_trans_node_B = y(p)) and (new_trans_node_B = trans_node_B)$ then $shorten_path_B := 0$, $shorten_path_BB := 0$, $test_node_B := y(p)$;
19. if $(new_trans_node_A = r(t)) and (new_trans_node_A > trans_node_A)$ then $shorten_path_AA := 1$, $test_node_A := r(t)$;
20. if $(new_trans_node_B = y(p)) and (new_trans_node_B > trans_node_B)$ then $shorten_path_BB := 1$, $test_node_B := y(p)$;
21. if $test_node_A \geq r(t)$ then $back_color_A := 1$;
22. else $back_color_A := 0$;
23. if $test_node_B \geq y(p)$ then $back_color_B := 1$;
24. else $back_color_B := 0$;
25. if $test_node_A > r(t)$ then $chan_test_node_A := 1$;
26. else $chan_test_node_A := 0$;
27. if $test_node_B > y(p)$ then $chan_test_node_B := 1$;
28. else $chan_test_node_B := 0$;
29. if $(trans_node_A = r(t)) and (new_trans_node_A \neq 0)$ then $one_branch_A := 1$;
30. if $(trans_node_B = y(p)) and (new_trans_node_B \neq 0)$ then $one_branch_B := 1$;
31. if $(trans_node_A > r(t)) and (new_trans_node_A \neq 0)$ then $end_another_branch_A := 1$;

32. if $(trans_node_B > y(p))and(new_trans_node_B \neq 0)$ then $end_another_branch_B := 1$;
33. if $(q_B = 0)and(STOP_B = 1)$ then $stop_condition_A := 1$;
34. else $stop_condition_A := 0$;
35. if $(q_A = 0)and(STOP_A = 1)$ then $stop_condition_B := 1$;
36. else $stop_condition_B := 0$;
37. друк V_H, E_H.

Розглянемо процедури, що використовуються в алгоритмі.

LIST_PROCE S_A($x_1, x_2, \dots, x_l$) :
1. if $Mes = "ISTHM_A(x_1, x_2, \dots, x_l)"$ then ISTHM_A($x_1, x_2, \dots, x_l$);
2. if $Mes = "INV_A(x_1, x_2, \dots, x_l)"$ then INV_A $(x_1, x_2, \dots, x_l)$;
3. if $Mes = "FORWARD_A(x_1, x_2, ..., x_q)"$ then FORWARD_A($x_1, x_2, ..., x_q$) ;
4. if $Mes = "ASSIGN_A"$ then $ASSIGN_A(\ )$;
5. if $Mes = "BACK_A"$ then $BACK_A(\ )$;
6. if $Mes = "SHORT_PATH_A"$ then $SHORT_PATH_A(\ )$;
7. if $Mes = "SEND_NUMB_A(x_1, x_2)"$ then $SEND_NUMB_A(x_1, x_2)$;
8. if $Mes = "SELECT_NODE_A(x)"$ then $SELECT_NODE_A(x)$;
9. if $Mes = "RESET_TRANSIT_A"$ then $RESET_TRANSIT_A(\ )$;
10. if $Mes = "CANCEL_SHORT_PATH_A"$ then $CANCEL_SHORT_PATH_A(\ )$;
11. if $Mes = "FIRST_ISTHM_A"$ then $FIRST_ISTHM_A(\ )$;
12. if $Mes = "INIT_POSITION_A(x)"$ then $INIT_POSITION_A(x)$;
13. if $Mes = "MOVING_N_AREA_A(x)"$ then $MOVING_N_AREA_A(x)$;
14. if $Mes = "STOP_A"$ then $STOP_A(\ )$.

ISTHM_A($x_1, x_2, \dots, x_l$) :
1. $E_H := E_H \cup \{(A[r(t)], B[x_1]); \dots; (A[r(t)], B[x_l])\}$;
2. $q_B := q_B - l$;
3. $N_v_B := N_v_B \setminus \{x_1, x_2, ..., x_l\}$;
4. if $first_isthm_A = 0$ then $\min_n_A := \min\{x_1, x_2, ..., x_l\}$.

INV_A $(x_1, x_2, \dots, x_l)$:
1. $E_H := E_H \cup \{(A[r(t)], A[x_1]); \dots; (A[r(t)], A[x_l])\}$;
2. $q_A := q_A - l$;
3. $N_v_A := N_v_A \setminus \{x_1, x_2, ..., x_l\}$.

FORWARD_A($x_1, x_2, ..., x_q$) :
1. if $previous_back_A = 1$ then do
2. $N_v_A := N_v_A \setminus \{r(t)\}$;
3. $q_A := q_A - 1$;
4. $previous_back_A := 0$;
5. end do;
6. else do
7. $q_A := q_A + q$;

8. $N_v_A := N_v_A \cup \{x_1, x_2, ..., x_q\}$;
9. end do;
10. $ct_A := ct_A + 1$;
11. $t := t + 1$;
12. $r(t) := ct_A$;
13. $V_H := V_H \cup \{A[Сч_A]\}$;
14. $E_H := E_H \cup \{(A[r(t-1)], A[r(t)])\}$;
15. $mr_B := 0$;
16. if $chan_test_node_A = 1$ then $test_node_A := r(t)$.

$ASSIGN_A(\ )$: $mr_B := 1$.

$BACK_A(\ )$:
1. з $r(1),..., r(t)$ видаляється елемент $r(t)$; $t := t - 1$;
2. $previous_back_A := 1$.

$SHORT_PATH_A(\ )$: $shorten_path_A := 0$.
$SEND_NUMB_A(x_1, x_2)$: $\min_n_A := \min\{x_1, x_2\}$.
$SELECT_NODE_A(x)$: if $x = \min_n_A$ then $\min_match_A := 1$.
$RESET_TRANSIT_A(\ )$: $\min_match_A := 0$; $\min_n_A := 0$.

$CANCEL_SHORT_PATH_A(\ )$:
1. $shorten_path_AA := 0$;
2. $one_branch_A := 0$;
3. $konec_drug_vetki_A := 0$.

$FIRST_ISTHM_A(\ )$:
1. $trans_node_A := r(t)$;
2. $first_isthm_A := 1$;
3. $\min_match_A := 0$;
4. $\min_n_A := 0$.

$INIT_POSITION_A(x)$: if $\exists n \in N_v_B : x - n > 0$ then $direction_A := 1$.

$MOVING_N_AREA_A(x)$:
1. $ct_A := ct_A + 1$;
2. $t := 1$;
3. $r(t) := Сч_A$;
4. $V_H := V_H \cup \{A[ct_A]\}$;
5. $E_H := E_H \cup \{(A[x], A[r(t)])\}$;
6. $mr_B := 0$;
7. $STOP_A := 0$;
8. $trans_node_A := ct_A$;
9. $perv_per_B := 1$;
10. $naprav_A := 0$;

11. $q_B := q_B - 1$;
12. $N_v_B := N_v_B \setminus \{x\}$;
13. $new_trans_node_B := x$.

$STOP_A(\ )$:
1. $t := 0$;
2. $mr_A := 0$;
3. $trans_node_A := 0$;
4. $new_way_A := 0$;
5. $new_way_B := 0$;
6. $test_node_A := 0$;
7. $test_node_B := 0$;
8. $STOP_A := 1$;
9. $previous_back_A := 0$;
10. $shorten_path_BB := 0$.

Процедури роботи зі списком повідомлень від агента B, які не розглядаються нижче, аналогічні процедурам роботи зі списком повідомлень від агента A.

LIST_PROCE S_B($k_1, k_2, ..., k_l$):
1. if $Mes =$ "$RETURN_B$" then $RETURN_B(\ )$;
2. if $Mes =$ "$ISTHM_B(k_1, k_2, ..., k_l)$" then $ISTHM_B(k_1, k_2, ..., k_l)$;
3. if $Mes =$ "$INV_B(k_1, k_2, ..., k_l)$" then $INV_B(k_1, k_2, ..., k_l)$;
4. if $Mes =$ "$FORWARD_B(k_1, k_2, ..., k_g)$" then $FORWARD_B(k_1, k_2, ..., k_g)$;
5. if $Mes =$ "$BACK_B$" then $BACK_B(\ )$;
6. if $Mes =$ "$SHORT_PATH_B$" then $SHORT_PATH_B(\ )$;
7. if $Mes =$ "$SEND_NUMB_B(k)$" then $SEND_NUMB_B(k)$;
8. if $Mes =$ "$SELECT_NODE_B(k)$" then $SELECT_NODE_B(k)$;
9. if $Mes =$ "$RESET_TRANSIT_B$" then $RESET_TRANSIT_B(\ )$;
10. if $Mes =$ "$CANCEL_SHORT_PATH_B$" then $CANCEL_SHORT_PATH_B(\ )$;
11. if $Mes =$ "$FIRST_ISTHM_B$" then $FIRST_ISTHM_B(\ )$;
12. if $Mes =$ "$INIT_POSITION_B(k)$" then $INIT_POSITION_B(k)$;
13. if $Mes =$ "$MOVING_N_AREA_B(k)$" then $MOVING_N_AREA_B(k)$;
14. if $Mes =$ "$STOP_B$" then $STOP_B(\ )$.

$RETURN_B(\ )$:
1. $E_H := E_H \setminus \{y(p-1), y(p)\}$;
2. $V_H := V_H \setminus \{B[ct_B]\}$;
3. $ct_B := ct_B - 1$;
4. $p := p - 1$;
5. $y(p) := ct_B$;
6. $previous_back_B := 1$.

$ISTHM_B(k_1, k_2, ..., k_l)$:
1. $E_H := E_H \cup \{(B[y(p)], A[k_1]); ; \ldots \rightleftarrows\rightleftarrows; (B[y(p)], A[k_l])\}$;

2. $q_A := q_A - l$;
3. $N_v_A := N_v_A \setminus \{k_1, k_2, ..., k_l\}$;
4. $mr_A := 1$.
5. if $first_isthm_B = 0$ then $\min_n_B := \min\{k_1, k_2, ..., k_l\}$.

$BACK_B(\ )$:
1. з $y(1),..., y(p)$ видаляється елемент $y(p)$;
2. $p := p - 1$;
3. $mr_A := 1$;
4. $previous_back_B := 1$.

Розглянемо основні властивості алгоритму розпізнавання.

Процедури $FORWARD_A(x_1, x_2, ..., x_q)$, $FORWARD_B(k_1, k_2, ..., k_g)$, MOVING_N_AREA_A() та MOVING_N_AREA_B() виконуються агентом-експериментатором при відвідуванні агентами-дослідниками білих вершин досліджуваного графа G. Цими процедурами агент-експериментатор створює по одній новій вершині графа H. При одночасному попаданні агентів-дослідників в одну білу вершину процедурами $FORWARD_A(x_1, x_2, ..., x_q)$, $FORWARD_B(k_1, k_2, ..., k_g)$ буде створено дві нові вершини графа H. Тобто для однієї відвіданої агентами-дослідниками вершини графа G в пам'яті агента-експериментатора створиться дві вершини графа H. Для того, щоб виправити це, на наступному кроці агент B, процедурою $RETURN_B(\ )$, видалить вершину, створену ним на попередньому кроці. Таким чином, виконання описаного алгоритму індукує відображення $\varphi : V_G \to V_H$. Причому $\varphi(v) = ct_A$ (вершина v забарвлена в червоний колір) та (вершина s забарвлена в жовтий колір). Зазначене відображення є бієкцією, оскільки агенти-дослідники відвідують усі вершини і кожній вершині, при першому її відвідуванні агентом, надається єдиний номер.

При виконанні процедур $FORWARD_A(x_1, x_2, ..., x_q)$, $FORWARD_B(k_1, k_2, ..., k_g)$ агент-експериментатор розпізнає ребро дерева (v,u) і так нумерує вершину u, що ребру $(v,u) \in G$ однозначно відповідає ребро $(\varphi(v), \varphi(u)) \in H$. При виконанні процедур $\text{INV_A}(x_1, x_2, ..., x_l)$ чи $\text{INV_B}(k_1, k_2, ..., k_l)$ агент-експериментатор розпізнає зворотні ребра $(v,u) \in G$ і ставить їм у однозначну відповідність ребра $(\varphi(v), \varphi(u)) \in H$. При виконанні процедур $ISTHM_A(x_1, x_2, ..., x_l)$ чи $ISTHM_B(k_1, k_2, ..., k_l)$ агент-експериментатор розпізнає перешийки $(v,u) \in G$ і ставить їм у однозначну відповідність ребра $(\varphi(v), \varphi(u)) \in H$. Отже, φ є ізоморфізмом графа G на граф H.

Теорема. *Три агенти, виконавши алгоритм розпізнавання на графі, розпізнають граф, що розглядається, з точністю до ізоморфізму.*

Визначимо часову, ємнісну та комунікаційну складності алгоритму. З опису випливає, що у кожному кроці алгоритму червоний (жовтий) шлях – це простий шлях, що з'єднує початкову вершину v (s – у разі агента B) з номером $\varphi(v) = 1$ ($\varphi(s) = 1$) з вершиною u (z) з номером $\varphi(u) = ct_A$ ($\varphi(z) = ct_B$). Отже, загальна довжина червоного та жовтого шляху не перевищує n.

На одноразове виконання процедур зі звичайного режиму роботи агент-дослідник витрачає один хід. При одноразовому виконанні процедур з режиму розпізнавання зворотних ребер агенти-дослідники розпізнають не більше $n-2$ зворотних ребер, на що витрачають один хід. При одноразовому виконанні процедури з режиму розпізнавання перешийків агенти-дослідники розпізнають не більше $n-2$ перешийки, на що так само витрачають один хід. При

одночасному попаданні двох агентів-дослідників в одну білу вершину агент A не змінює режим роботи, а агент B витрачає один хід на повернення в свій підграф для розпізнавання. На виконання кожної процедури під час пошуку нової території для розпізнавання агент-дослідник витрачає один хід. При розрахунку часової складності алгоритму вважатимемо, що ініціалізація алгоритму, аналіз околу $Q(v)$ робочої вершини та вибір однієї з можливих процедур займають певну постійну кількість одиниць часу. Також будемо вважати, що вибір ребер, прохід по них агента-дослідника та обробка повідомлень, відправлених на даному етапі, здійснюється за 1 одиницю часу. Тоді часова складність алгоритму визначається наступним чином.

Процедури зі звичайного режиму роботи виконуються не більше $2\times(n-1)+4\times n$ раз та загальний час їх виконання оцінюється як $O(n)$. Процедури, пов'язані з режимом розпізнавання зворотних ребер, виконуються не більше $n\times(n-2)$ разів, тобто загальний час виконання процедур з режиму розпізнавання зворотних ребер оцінюється як $O(n^2)$. Процедури, пов'язані з режимом розпізнавання перешийків, виконуються не більше, ніж $n\times(n-2)+(n-1)$ раз з урахуванням запиту змінних та загальний час їх виконання оцінюється як $O(n^2)$. Час простою агентів-дослідників у всіх режимах роботи не перевищує $3\times(n-1)+2\times(n-2)+4\times n$, тобто оцінюється зверху як $O(n)$. Час виконання процедур агентів-дослідників для переходу до нових областей розпізнавання оцінюється як $O(n)$. Отже, сумарна часова складність $T(n)$ алгоритму задовольняє співвідношенню: $T(n)=O(n^2)$. А верхня оцінка числа переходів по ребрах, що здійснюються агентом-дослідником оцінюється зверху як $O(n)+O(n)=O(n)$.

Ємна складність $S(n)$ алгоритму визначається складністю списків V_H, E_H, $r(1),...,r(t)$, $y(1),...,y(p)$, N_v_A та N_v_B, складність яких відповідно визначається величинами $O(n)$, $O(n^2)$, $O(n)$, $O(n)$, $O(n^2)$, $O(n^2)$. Отже: $S(n)=O(n^2)$.

Комунікаційна складність $K(n)$ алгоритму визначається обсягом інформації, якою необхідно обмінятися агентам для розпізнавання графа. При роботі в звичайному режимі роботи обсяг переданої агентами-дослідниками інформації оцінюється як $O(n^2)$. Агент-експериментатор, при цьому передасть обсяг інформації, що оцінюється як $O(n)$. При роботі агентів в режимі розпізнавання зворотних ребер і в режимі розпізнавання перешийків обсяг переданої інформації оцінюється як $2\cdot O(n^2)$. Отже, сумарна комунікаційна складність $K(n)$ алгоритму задовольняє співвідношення: $K(n)=O(n^2)$.

Теорема. *Часова складність алгоритму розпізнавання дорівнює* $O(n^2)$, *ємнісна –* $O(n^2)$, *комунікаційна –* $O(n^2)$. *Число переходів по ребрах, які треба здійснити агентам-дослідникам, оцінюється як* $O(n)$. *При цьому алгоритм використовує 3 фарби.*

Висновки

В роботі запропоновано новий алгоритм розпізнавання скінчених неорієнтованих графів часової складності $O(n^2)$, ємнісної складності $O(n^2)$ та комунікаційної складності $O(n^2)$. Число переходів по ребрах, які треба здійснити агентам-дослідникам для розпізнавання графа за даним алгоритмом, оцінюється як $O(n)$. В роботі наведено процедуру оптимізації розбиття графа на частини, що розпізнаються різними агентами-дослідниками. Агенти-дослідники мають скінчену на кожному кроці, але необмежено зростаючу пам'ять і використовують по дві фарби кожен (всього три фарби різного кольору).

Література

1. Albers S., Henzinger M. R. *Exploring unknown environments*. SIAM Journal on Computing, 2000. 29 (4). P. 1164–1188.

2. Dudek G., Jenkin M., Milios E., Wilkes D. *Map validation in a graphlike world*. Proceedings of the 13th International Joint Conference on Artifical Intelligence. San Fransisco: Morgan Kaufmann Publishers Inc., 1993. P. 1648–1653.

3. Стёпкин А. В. *Распознавание конечных графов тремя агентами.* Искусственный интеллект, 2011. №2. С. 84 – 93.

4. Nagavarapu S. C., Vachhani L., Sinha A. et al. *Generalizing Multi-agent Graph Exploration Techniques*. International Journal of Control, Automation and Systems. 2020. https://doi.org/10.1007/s12555-019-0067-8

5. Dey, S.; Xu, H. *Intelligent Distributed Swarm Control for Large-Scale Multi-UAV Systems: A Hierarchical Learning Approach*. Electronics 2023, 12(1):89. https://doi.org/10.3390/electronics12010089

6. Dongyu Li, Shuzhi Sam Ge, Wei He, Guangfu Ma, Lihua Xie. *Multilayer formation control of multi-agent systems*. Automatica. V 109. 2019 https://doi.org/10.1016/j.automatica.2019.108558

7. Amirkhani, A., Barshooi, A.H. *Consensus in multi-agent systems: a review*. Artif Intell Rev 55, 3897–3935 (2022). https://doi.org/10.1007/s10462-021-10097-x

8. Shannon C. E. *Presentation of a maze-solving machine*. Cybernetics Trans, of the 8 th Conf. of the JosiahMacy Jr. Found / Editor: H. Foerster. 1951. P. 173–180.

9. Dopp K. *Automaten in labirinthen*. Electronische Informationsverarbeitung und Kybernetik. 1971. V.7, №2. P. 79–94.

10. Dudek G., Jenkin M., Milios E., Wilkes D. *Topological exploration with multiple robots*. Robotics with Application (ISORA): Proc. 7th International Symposium. Alaska, 1998

11. Wang H., Jenkin M., Dymond P. *It can be beneficial to be 'lazy' when exploring graph-like worlds with multiple robots*. Advances in Computer Science and Engineering (ACSE): In Proceedings of the IASTED International Conference. 2009. P. 55–60.

12. Zhang C. *Parallelizing Depth-First Search for Robotic Graph Exploration*. Harvard College, Cambridge, Massachusetts. 2010.

13. Nagavarapu, S. C., Vachhani, L. & Sinha, A. *Multi-Robot Graph Exploration and Map Building with Collision Avoidance: A Decentralized Approach*. Journal of Intelligent & Robotic Systems. 2016. 83. P. 503–523 https://doi.org/10.1007/s10846-015-0309-9

14. Das S., Flocchini P., Kutten S., Nayak A., Santoro N. *Map construction of unknown graphs by multiple agents*. Theoretical Computer Science. 2007. V.385, 1–3. P. 34 – 48.

15. Stepkin A. *Using a Collective of Agents for Exploration of Undirected Graphs*. Cybernetics and Systems Analysis. V.51, 2. 2015 223–233. https://doi.org/10.1007/s10559-015-9715-z

References

1. Albers S., Henzinger M. R. *Exploring unknown environments*. SIAM Journal on Computing, 2000. 29 (4). P. 1164–1188.

2. Dudek G., Jenkin M., Milios E., Wilkes D. *Map validation in a graphlike world*. Proceedings of the 13th International Joint Conference on Artifical Intelligence. San Fransisco: Morgan Kaufmann Publishers Inc., 1993. P. 1648–1653.

3. Stepkin A. V. Raspoznavanie konechnykh hrafov tremya ahentami. Iskusstvennyy intellekt, 2011. #2. S. 84 – 93.

4. Nagavarapu S. C., Vachhani L., Sinha A. et al. *Generalizing Multi-agent Graph Exploration Techniques*. International Journal of Control, Automation and Systems. 2020. https://doi.org/10.1007/s12555-019-0067-8

5. Dey, S., Xu, H. *Intelligent Distributed Swarm Control for Large-Scale Multi-UAV Systems: A Hierarchical Learning Approach*. Electronics 2023, 12(1):89. https://doi.org/10.3390/electronics12010089

6. Dongyu Li, Shuzhi Sam Ge, Wei He, Guangfu Ma, Lihua Xie. Multilayer formation control of multi-agent systems. Automatica. V 109. 2019 https://doi.org/10.1016/j.automatica.2019.108558

7. Amirkhani, A., Barshooi, A. H. Consensus in multi-agent systems: a review. Artif Intell Rev 55, 3897–3935 (2022). https://doi.org/10.1007/s10462-021-10097-x

8. Shannon C. E. Presentation of a maze-solving machine. Cybernetics Trans, of the 8 th Conf. of the JosiahMacy Jr. Found / Editor: H. Foerster. 1951. P. 173–180.

9. Dopp K. Automaten in labirinthen. Electronische Informationsverarbeitung und Kybernetik. 1971. V.7, №2. P. 79–94.

10. Dudek G., Jenkin M., Milios E., Wilkes D. *Topological exploration with multiple robots*. Robotics with Application (ISORA): Proc. 7th International Symposium. Alaska, 1998

11. Wang H., Jenkin M., Dymond P. *It can be beneficial to be 'lazy' when exploring graph-like worlds with multiple robots*. Advances in Computer Science and Engineering (ACSE) : In Proceedings of the IASTED International Conference. 2009. P. 55–60.

12. Zhang C. *Parallelizing Depth-First Search for Robotic Graph Exploration*. Harvard College, Cambridge, Massachusetts. 2010.

13. Nagavarapu, S. C., Vachhani, L. & Sinha, A. *Multi-Robot Graph Exploration and Map Building with Collision Avoidance: A Decentralized Approach*. Journal of Intelligent & Robotic Systems. 2016. 83. P. 503–523 https://doi.org/10.1007/s10846-015-0309-9

14. Das S., Flocchini P., Kutten S., Nayak A., Santoro N. *Map construction of unknown graphs by multiple agents*. Theoretical Computer Science. 2007. V.385, 1–3. P. 34 – 48.

15. Stepkin A. *Using a Collective of Agents for Exploration of Undirected Graphs*. Cybernetics and Systems Analysis. V.51, 2. 2015 223–233. https://doi.org/10.1007/s10559-015-9715-z

The article has been sent to the editors 22.12.24.
After processing 02.02.25.
Submitted for printing 30.03.25.